MICROPROCESSOR TECHNOLOGY

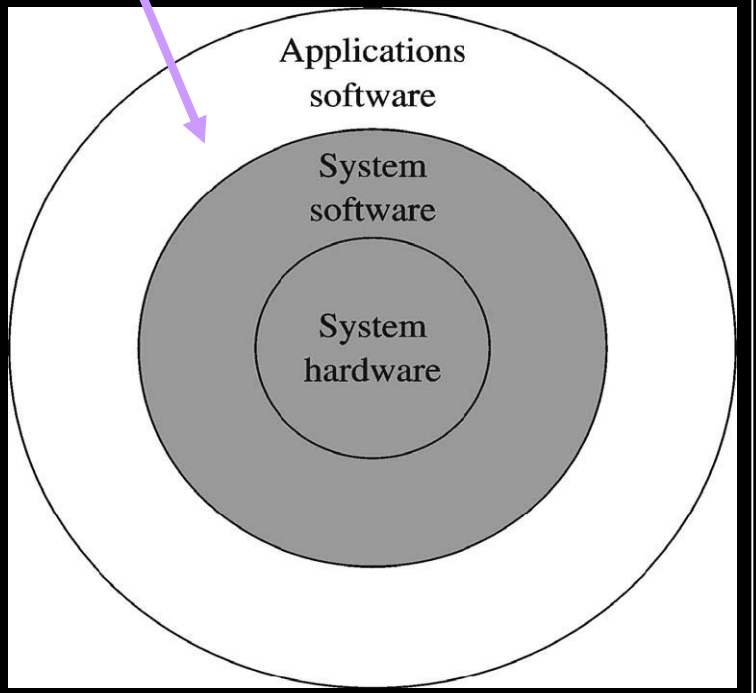
Dr. Hossam El-Din Moustafa

Lecture 4

Ch.2 A Top-Level View of Computer Function

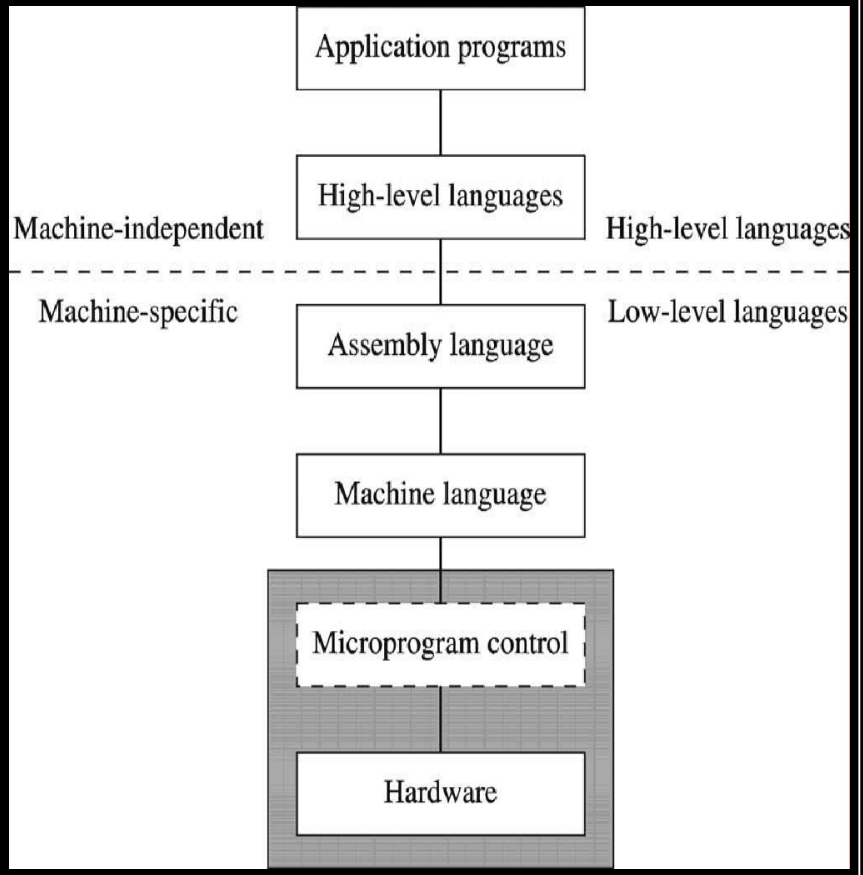
The User's View

- A typical user can only see solid components of the system.
- He is not concerned with the deep construction of the system or how it works.
- He uses his keyboard and mouse to enter some data to the application he is using and gets some piece of result using his monitor, printer, or speaker no matter what happens inside the computer.



The Programmer's View

- The programmer's view differs according to the type of programming he is doing.
- Programming languages are divided into two main groups; ***low-level*** and ***high-level*** languages



Low-Level Language

- Machine specific or dependent
- Consists of streams of 0's and 1's and usually represented in computer memory using hexadecimal system.

  Assembly language is slightly higher than M/C language. It is also human readable.
- One-to-one correspondence with most M/C language instructions

Low-Level Language (*cont.*)

- Uses assembler to convert assembly code into M/C code
- Large number of instructions
- Program maintenance is difficult
- Program development is slow
- Code is not portable

High-Level Language

- Machine independent
- Human readable language that can be recognized by human's English language.
- Each instruction corresponds to a complete block of M/C language code

High-Level Language

- Uses compilers to convert the high level language code into M/C code directly or indirectly via an assembler
- Relatively fewer instructions
- Program maintenance is easy
- Program development is faster
- Code is partially portable

Readability of Assembly Language

- Readability of assembly language instructions is much better than the machine language instructions
  - » Machine language instructions are a sequence of 1s and 0s

Assembly Language		Machine Language (in Hex)
inc	result	FF060A00
mov	class_size,45	C7060C002D00
and	mask,128	80260E0080
add	marks,10	83060F000A

Assembly Vs. C language

- **Some simple high-level instructions can be expressed by a single assembly instruction.**
- ***Example:***

inc result	Result++;
mov size,45	Size = 45;
and mm, 128	mm &= 128;
add ll,10	ll += 10;

Assembly Vs. C language

- Most high-level language instructions need more than one assembly instruction

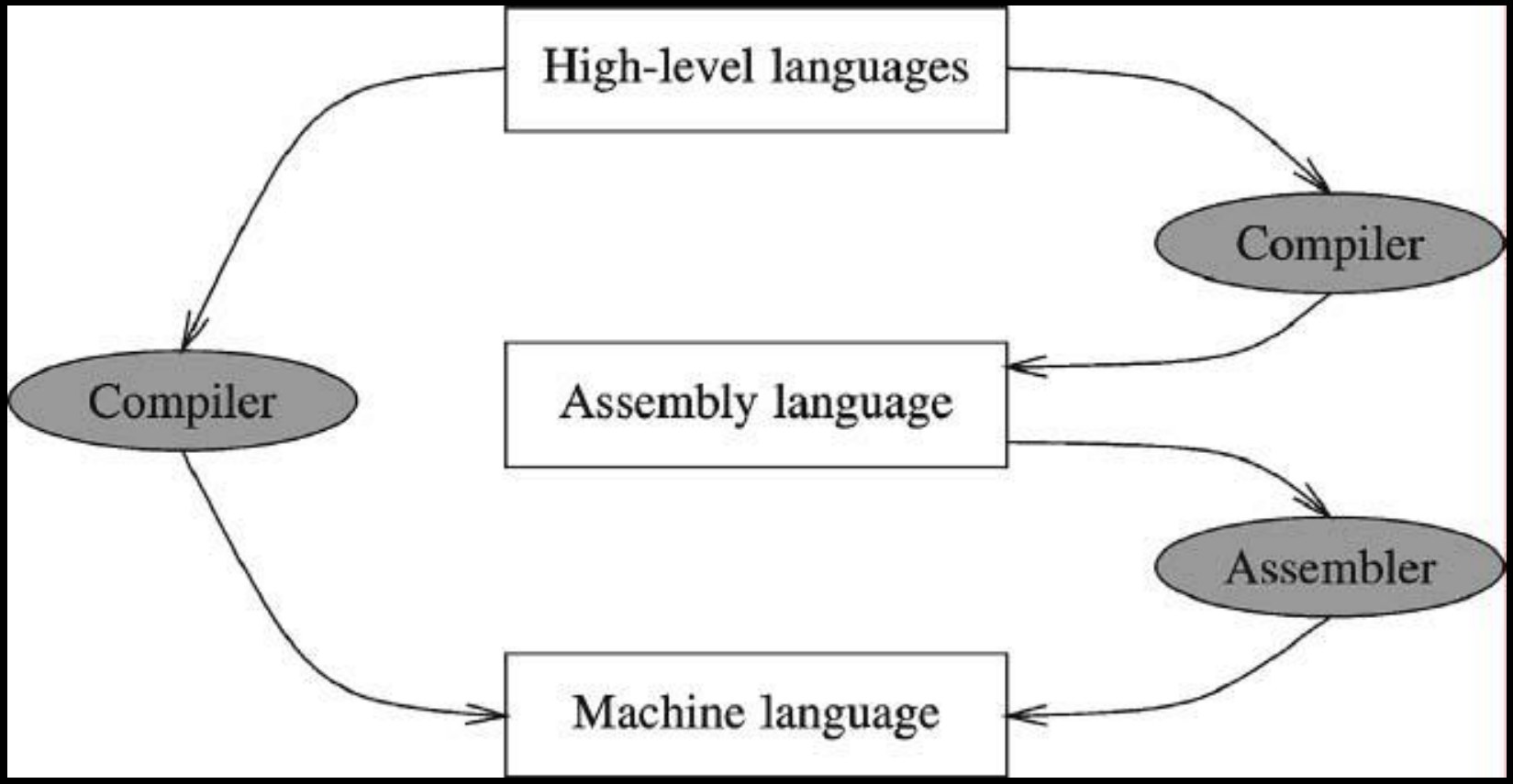
C	Assembly Language
`size = value;`	`mov AX,value` `mov size,AX`

`sum += x + y + z;`	`mov AX,sum`
	`add AX,x`
	`add AX,y`
	`add AX,z`
	`mov sum,AX`

Assembler Vs. Compiler

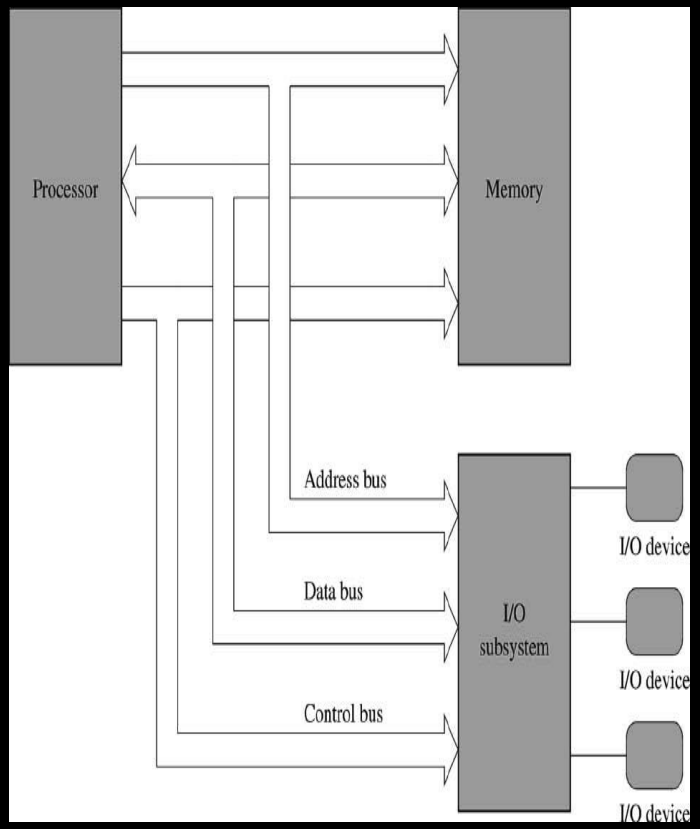
- ***The Assembler:*** **It is a program written to convert assembly instructions into its corresponding M/C code. Some assembly instructions have a one-to-one correspondence relationship, others have one-to-many correspondence.**
- ***The Compiler:*** **It is a program written to convert high level language code into its corresponding M/C code. Some compilers convert the code directly into M/C code while the others convert the code into assembly code, then uses an assembler to convert the generated assembly code into M/C code**

Assembler Vs. Compiler



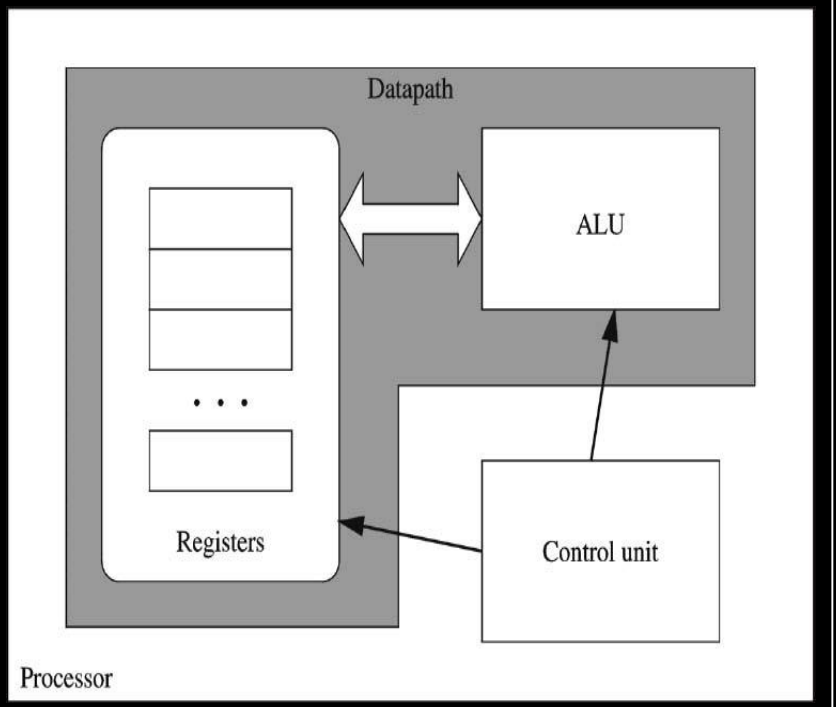
The Architect's View

- The architect is concerned with the functional issues with no interest with the internal or low-level design issues.
- He uses a higher level building block to express the desired construction.
- He sees the system as microprocessor, RAM, I/O devices, connected together by buses.



The Implementer's View

- This is the person who is responsible for implementing the designs generated by the architect.
- He is also concerned with the internal and the low level design.
- He sees the microprocessor as a control unit, an ALU, and some registers.
- He sees the RAM as a group of general purpose registers



The Implementer's View

ALU (Arithmetic and Logic Unit):

- **It is responsible for the arithmetic and logic operations as it is obvious from its name.**
- **It only deals with the inner data storage cells of the microprocessor (registers).**
- **It has no means of communication with external memory.**
- **So, the inputs must be in the registers and the results will be put back to the registers.**

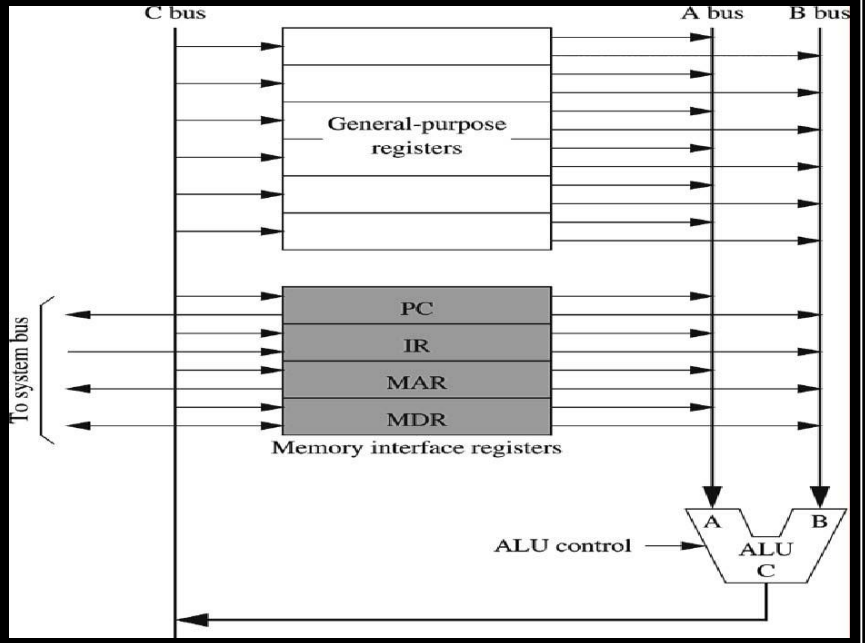
The Implementer's View

Registers:

- These are the memory units within the CPU.
- They are used to hold the inputs to the ALU and also to hold the intermediate and final results.
- They contain two types of registers; ***memory interface registers*** and ***general purpose registers***.
- ***Memory interface registers*** are used for the execution phase of an instruction.
- ***General purpose registers*** are used for holding intermediate results while processing instructions.

The Implementer's View

- *PC* is the Program Counter register which contains the address of the next instruction in memory to be fetched.
- *IR* is the Instruction Register which holds the instruction in progress.
- *MDR* is the Memory Address Register which holds the *data* to be processed.
- *MAR* is the register which holds the *address* of the data block in memory to be processed.



The Implementer's View

- ***Control Unit:***
- **It is responsible for issuing control signals to all components of the computer system and deciding which system component can have access to another and when this can happen.**
- **They are connected to the inputs of the ALU through two internal buses (A, B).**
- **The output of the ALU is connected again to the registers input through a third bus (C).**
- **Using these three buses, data stored inside different registers can be processed and stored back again inside the registers.**

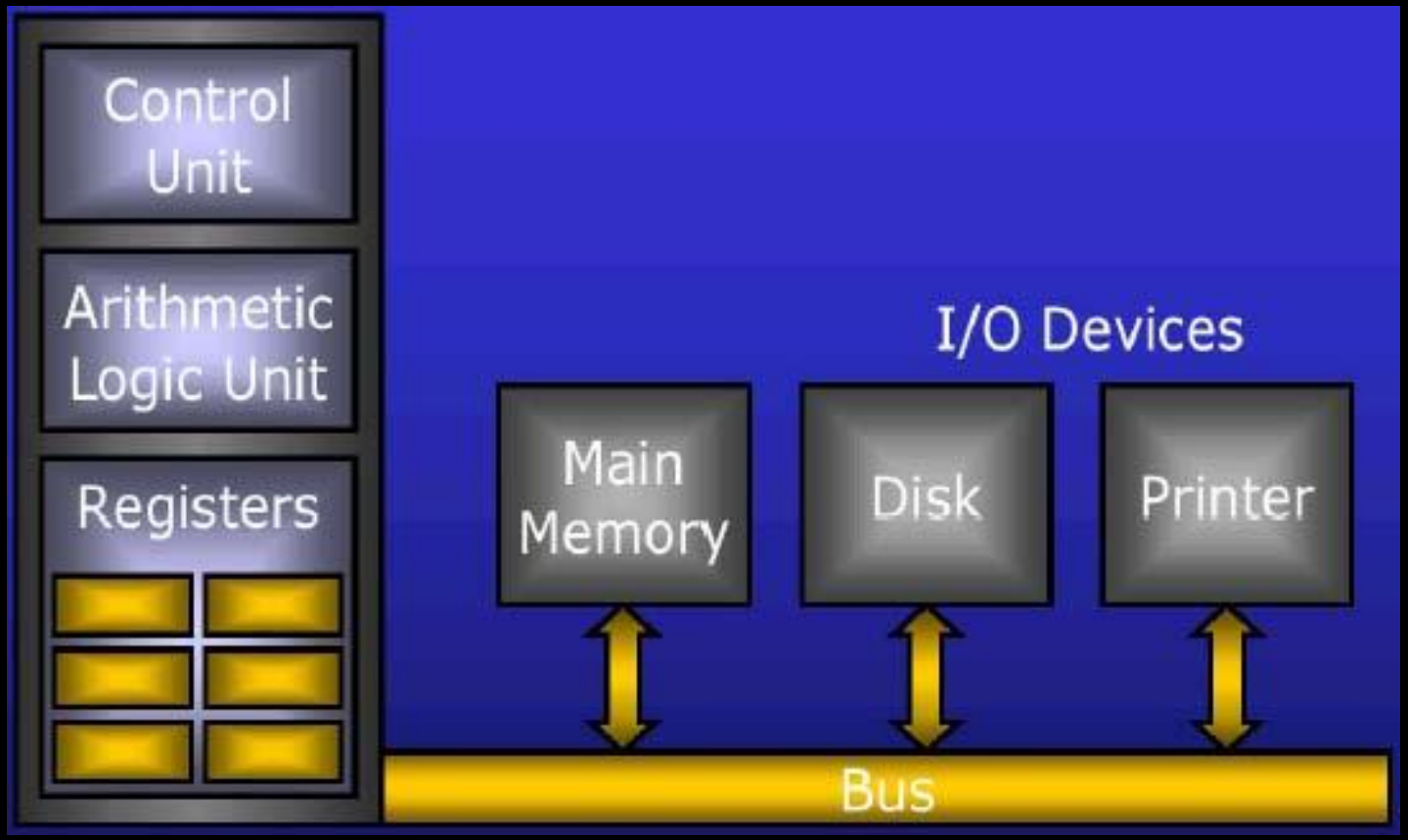
Von Neumann Architecture (Architect's View)

Every general-purpose computer has four basic hardware components:

1) Central Processing Unit (CPU)
2) Main memory
3) Input device
4) Output device

These components are connected to each other by **the bus.**

Von Neumann Architecture (A Typical Microprocessor Layout)

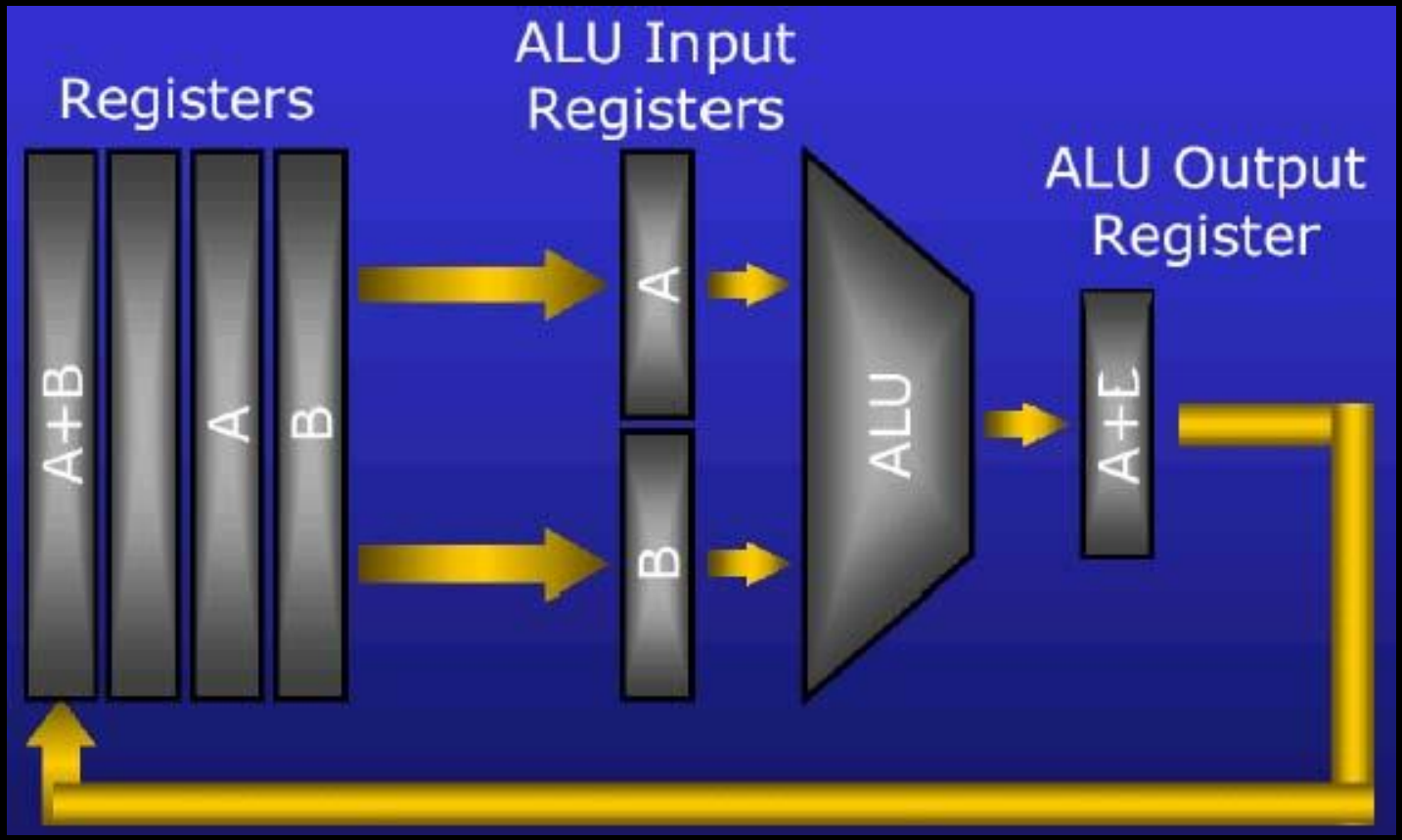


The Execution Cycle

- To execute a program (a sequence of instructions) stored in main memory, the CPU controller performs a ***fetch-execute cycle***:

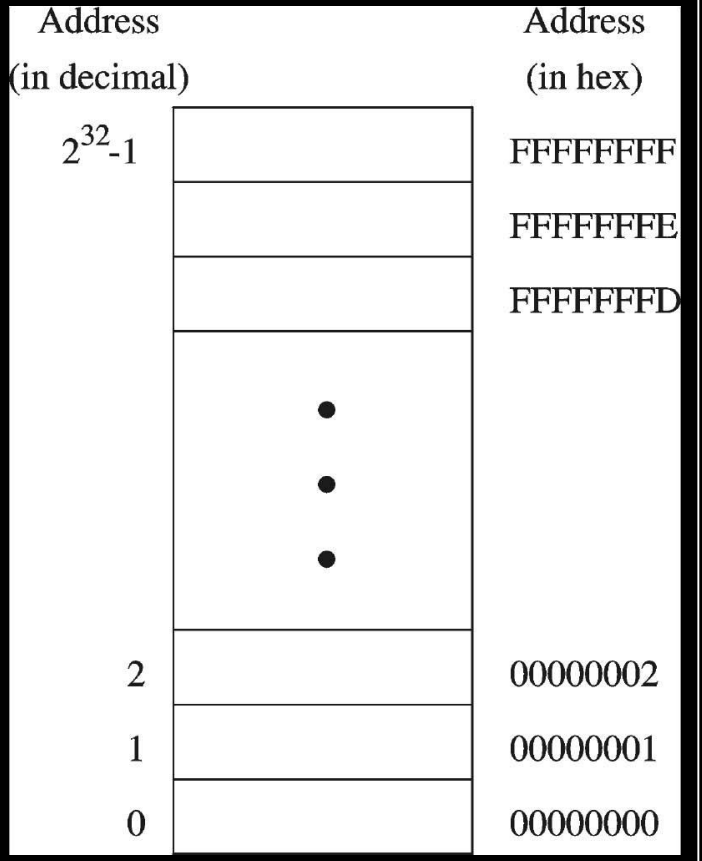
1. Fetch an instruction from the memory.
2. Decode the instruction.
3. Execute the instruction by sending control signals to the ALU and other parts of the computer.
4. Send the result back to memory, or to an input/output device.

Von Neumann Architecture (The Data Bus)



The Main Memory

- It is a set of numbered cells.
- The number is called address, which can be used to locate the cell.
- Each cell contains data, a number, which is the thing to be processed by the computer.
- The memory also contains the program, coded as numbers, which describe how to process data.



The Main Memory

- Therefore, inside the machine, there is nothing to distinguish data from programs, nor data from addresses, except in how they are used.
- All commercial computers are based on *stored program* concept, with programs and data sharing the same main memory.
- Such computers are called Von Neumann machines.

The Bus

- It provides a mean for the connection /communication among CPU, main memory, and input/output (I/O) devices.
- During the execution of a program, there are ***three*** sorts of data flowing along the bus: addresses, instructions and actual data, and control signals.

The Bus

A normal bus includes three parts:

- **Address bus:** for passing the *addresses* of instructions and data in memory.
- ***Data bus:*** for transferring the instructions and data themselves.
- ***Control bus:*** for issuing control signals such as reading or writing.

The size of each bus affects the performance of the system, determines the type of microprocessor and limits the maximum size of memory that can be addressed.

(Details will be presented in the next lecture)

Input and Output Devices

- I/O operation moves data between the computer and its external environment.
- Input devices include keyboard, disk drives, CD and DVD drives, magnetic tape drives, mouse devices, scanners, modems, microphones, stylus, blue-tooth, etc.
- Output devices include disk drives, printers, screens, CD and DVD drives, magnetic tape drives, modems, speakers, etc.
- Communications with these devices can be done in a manner similar to memory reading/writing.

Thank You

With all best wishes !!